

Resolution Simulates Ordered Binary Decision Diagrams for Formulas in Conjunctive Normal Form

Olga Tveretina

School of Computer Science
University of Hertfordshire
United Kingdom
o.tveretina@herts.ac.uk

Abstract. A classical question of propositional logic is one of the shortest proof of a tautology. A related fundamental problem is to determine the relative efficiency of standard proof systems, where the relative complexity is measured using the notion of polynomial simulation.

Presently, the state-of-the-art satisfiability algorithms are based on resolution in combination with search. An Ordered Binary Decision Diagram (OBDD) is a data structure that is used to represent Boolean functions.

Groote and Zantema have proved that there is exponential separation between resolution and a proof system based on limited OBDD derivations. However, formal comparison of these methods is not straightforward because OBDDs work on arbitrary formulas, whereas resolution can only be applied to formulas in Conjunctive Normal Form (CNFs).

Contrary to popular belief, we argue that resolution simulates OBDDs polynomially if we limit both to CNFs and thus answer negatively the open question of Groote and Zantema whether there exist unsatisfiable CNFs having polynomial OBDD refutations and requiring exponentially long resolution refutations.

1 Introduction

Propositional proof complexity is the study of the lengths of proofs of statements expressed as propositional formulas. It is tightly connected in many ways to computational complexity, classical proof theory and practical questions of automated deduction.

A classical question of propositional proof complexity is one of the shortest proof of a tautology. A related fundamental problem is to determine the relative efficiency of standard proof systems, where the relative complexity is measured using the notion of polynomial simulation.

Proof systems. Propositional proof systems were defined by Cook and Reckhow as polynomial-time functions which have as their range the set of all tautologies [6]. They also noticed that if there is no propositional proof system that admits proofs polynomial in size of the input formula then the complexity classes NP and co-NP are different, and hence $P \neq NP$.

In [2], Atserias, Kolaitis and Vardi generalised the notion of a refutational propositional proof system, viewing it as a special case of constraint propagation. Their proof system consists of the following four rules: (1) **Axiom** defines the initial set of constraints; (2) **Join** combines two constraints by intersecting two relations and extending them to all variables occurring in either one of them; (3) **Projection** computes the projection of a constraint which is the existential quantification; (4) **Weakening** relaxes the constraint by enlarging its relation.

This generalisation brings the methods of constraint propagation to the area of proof complexity. On the other hand it introduces new classes of proof systems. The existing refutational proof systems can be viewed as a special case of this constraint propagation system.

Efficiency of proof systems. One of the most fundamental problems in the area of propositional proof complexity is to determine the relative efficiency of standard proof systems as it has been introduced by Cook and Reckhow in [6] who found it useful to separate the idea of providing a proof from that being efficient.

Proof systems are compared according to their strength using the notion of polynomial simulation. A proof system S_1 simulates polynomially a proof system S_2 if every tautology has proofs in S_1 of size at most polynomially larger than in S_2 . Proof systems S_1 and S_2 are equivalent if they simulate each other polynomially.

Although substantial progress has been made in determining the relative complexity of proof systems and in proving strong lower bounds for some relatively weak proof systems, some major problems still remain unsolved.

As it is defined by Razborov in [14], the question of existence of an efficient proof has to be separated from another important question how to find such a proof efficiently and whether this search adds substantially to the inherent complexity of finding the shortest proof in a specific proof system. It is formalised by the notion of automatizability: a proof system is automatizable if it produces a proof of a tautology in time polynomial in the size of its smallest proof [3].

The current proof systems of practical use are not automatizable (or just weakly automatizable). That is why in addition to lower bounds, there is practical interest in understanding relative efficiency of somewhat weaker proof systems. Interesting examples of such systems are those based either on general resolution or on classical OBDDs not utilising existential quantification.

Resolution versus OBDDs. In the automated reasoning community resolution and OBDDs are popular techniques for solving the propositional satisfiability problem abbreviated as SAT. In fact, both resolution and OBDDs are families of algorithms, where each corresponds to a specific way of making choices.

Resolution underlies the vast majority of all proof search techniques in this area. For example, the DPLL algorithm [7], as well as the clause learning methods are highly optimised implementations of resolution [12]. It has been shown in [13] that modern SAT solvers simulate resolution polynomially.

An OBDD is a canonical data structure that is used for the symbolic representation of Boolean functions [5,19]. Atserias et al. introduced and studied a proof system operating with OBDDs as a special case of constraint propagation [2]. They proved that OBDD based refutations polynomially simulate resolution if they utilise existential quantification. That is OBDD based proof systems containing all four rules Axiom, Join, Weakening and Projection are strictly stronger than resolution but they are still exponential [9].

In the following we consider the OBDD proof system which contains just two rules, Axiom and Join. These rules are equivalent to the Apply operator as it is defined in [5].

Benchmark studies show incomparable behaviour of resolution and such OBDD based systems [18]. Groote and Zantema proved that resolution and OBDDs do not simulate each other polynomially on arbitrary inputs for limited OBDD derivations [8]. Tveretina, Sinz and Zantema strengthened the above result and presented a class of CNFs hard for an arbitrary OBDD derivation and easy for resolution [17].

In general, formal comparison of resolution and OBDDs is not straightforward because the later work on arbitrary formulas, whereas resolution can take as an input only CNFs.

We argue that resolution simulates OBDDs polynomially if we limit both to CNFs. Thus we answer negatively the open question of Groote and Zantema posed in [8] whether there exist unsatisfiable CNFs having polynomial OBDD refutations and requiring exponentially long resolution refutations.

Previous work. There are several works which study the relative efficiency of resolution based and OBDD based proof systems. The most relevant studies to our setting are the following ones.

Peltier shows in [11] that resolution augmented with the extension rule polynomially simulates OBDDs in the following sense: for any unsatisfiable formula φ there exists a refutation of φ with the size polynomially bounded by the maximal size of the reduced OBDDs corresponding to the subformulas occurring in φ . As mentioned before, Atserias et al prove in [2] that OBDD based refutations utilising existential quantification polynomially simulate resolution; moreover they are exponentially stronger. Groote and Zantema construct in [8] biconditional formulas that have short OBDD refutations and after transforming them into CNFs they require exponentially long resolution proofs. But the same formulas after transformation into CNFs have exponentially long OBDD proofs.

Main result. We show that for any unsatisfiable CNF φ there exists a resolution refutation of φ with the size polynomially bounded by the size of an OBDD based refutation of φ if it consists of two rules **Axiom** and **Join** and uses two standard reduction rules, elimination and merging. We now formally state the theorem.

Theorem 1. *Assume an unsatisfiable CNF φ . If there is an OBDD refutation of φ with two rules **Axiom** and **Join** of size n then there is a resolution refutation of φ of size $O(n^2)$.*

Our main argument is based on the idea that the elimination rule can be simulated by applying the resolution rule on the variable corresponding to the eliminated node [11]. But we use it differently.

We strengthen this idea and prove that the number of resolution steps corresponding to the elimination of a node is bounded by the number of clauses in the input CNF φ . Moreover, we show that it is an invariant property: although resolution steps generate new clauses, the number of resolution steps needed to simulate elimination of a node in an intermediate OBDD remains bounded by the number of clauses of φ encoded by this OBDD. Furthermore, we show that it is sufficient to simulate only the elimination rule, that is the merging rule plays no role in the context.

The remainder of the paper. We give the necessary background in Section 2. In Section 3 we introduce two proof systems of interest: one is based on resolution and the other corresponds to OBDD derivations based on the **Axiom** and **Join** rules. In Section 4 we show how to simulate the elimination rule using resolution and in Section 5 we prove our main result. Section 6 contains concluding remarks.

2 Preliminaries

2.1 Propositional Logic and Conjunctive Normal Forms

In this section we recall some basic notations about propositional logic and only provide a short overview of the main definitions.

In the following we consider propositional formulas in Conjunctive Normal Form (CNF) built using variables from a set **var**. A literal l is either a variable x or its negation $\neg x$ with $\text{var}(l) = x$. A clause C is a disjunction of literals, and a CNF φ is a conjunction of clauses. By **Cnf** we denote the set of all CNFs.

We define $\text{Cls}(\varphi)$ to be the set of clauses, $\text{Lit}(\varphi)$ the set of literals, and $\text{var}(\varphi)$ the set of variables contained in the CNF φ .

We use $\varphi|_l$ to denote the CNF obtained from φ by deleting all clauses containing a literal l and removing $\neg l$ from the rest of the clauses. Note that $\varphi = \varphi|_l$ if $l \notin \text{Lit}(\varphi)$.

A truth assignment is a function $A : \text{var} \rightarrow \{\text{true}, \text{false}\}$. We denote by \mathcal{A} the set of all possible assignments. The truth values of literals, clauses and CNFs are defined in a standard way.

We write $A \models \varphi$ if φ evaluates to **true** for the assignment A , otherwise we write $A \not\models \varphi$. We say that φ is unsatisfiable if $A \not\models \varphi$ for any $A \in \mathcal{A}$, otherwise it is satisfiable; φ is a tautology if $A \models \varphi$ for any $A \in \mathcal{A}$.

We say that two CNFs φ and ψ are logically equivalent, denoted $\varphi \equiv \psi$, if $A \models \varphi$ if and only if $A \models \psi$ for any $A \in \mathcal{A}$.

We use \top for the empty set of clauses and \perp for the CNF consisting of the empty clause. By definition the empty clause is unsatisfiable that is it is equivalent to **false**, and the the empty set of clauses is equivalent to **true**.

2.2 Ordered Binary Decision Diagrams

The concept of ordered binary decision diagrams (OBDDs) was first proposed by Lee in [10] as a means to represent propositional formulas (Boolean functions) compactly as directed acyclic graphs (DAGs). Then it was further developed to a data structure by Acers [1] and Boute [4], and subsequently by Bryant [5].

Definition 1 (An OBDD). An OBDD B is a directed acyclic graph satisfying the following:

1. it has a unique node called the root and denoted by $\text{root}(B)$;
2. each inner node p is labeled by the propositional variable $\text{var}(p)$ and has exactly two successors, a **false**-successor and a **true**-successor;
3. the inner nodes build the set $\text{Node}(B)$, and the labels build the set $\text{var}(B)$;
4. each leaf node is labeled by either **true** or **false**;
5. there is a total variable order \prec such that for each transition from the inner node with label x to the inner node with label y we have that $x \prec y$.

We use $\text{high}(p)$ and $\text{low}(p)$ to denote the OBDDs rooted at the **true**-successor and the **false**-successor of p ; $|B|$ is the size of B , that is the number of its inner nodes.

We use $B_1 \equiv B_2$ to denote that B_1 and B_2 are isomorphic OBDDs defined as follows:

- both B_1 and B_2 consist either of the node **true** or of the node **false**;
- $\text{var}(\text{root}(B_1)) = \text{var}(\text{root}(B_2))$, $\text{high}(B_1) \equiv \text{high}(B_2)$ and $\text{low}(B_1) \equiv \text{low}(B_2)$.

OBDD operations are applicable only to OBDDs that respect the same variable ordering. To shorten the notations in the rest of the paper, we assume without explicitly stating that all the variables agree on the common variable order $x_1 \prec x_2 \prec x_3 \prec \dots$ when considering different OBDDs in the same context.

Definition 2 (Path). A *path* of an OBDD B is a sequence $\alpha = l_1 \dots l_k$ of literals with $k \geq 1$ such that there are $p_1, \dots, p_k \in \text{Node}(B)$, where

- $p_1 = \text{root}(B)$;
- for $1 \leq i < k$, either $p_{i+1} = \text{root}(\text{high}(p_i))$ and $l_i = \text{var}(p_i)$, or $p_{i+1} = \text{root}(\text{low}(p_i))$ and $l_i = \neg \text{var}(p_i)$;

- $\text{high}(p_k) \in \{\text{false}, \text{true}\}$ if $l_k = \text{var}(p_k)$;
- $\text{low}(p_k) \in \{\text{false}, \text{true}\}$ if $l_k = \neg \text{var}(p_k)$.

We use $\text{Path}(\mathbf{B})$ to denote the set of all paths of \mathbf{B} , and $\text{Path}^f(\mathbf{B})$ to denote the set of all paths that go to the false node. By $\text{Path}(p)$ we mean the set of all paths that go through the inner node p , and $\text{Path}^f(p) = \text{Path}(p) \cap \text{Path}^f(\mathbf{B})$.

A path can be seen as a conjunction of literals. In this way, each path $\alpha = l_1 \dots l_k$ naturally induces the set $\mathcal{A}(\alpha)$ of truth assignments evaluating each l_i to true. We write $\alpha \not\models C$ for a clause C if $A \not\models C$ for any $A \in \mathcal{A}(\alpha)$. Where it is convenient, we see a path as a set of literals and use the set notations.

We say that a CNF φ and an OBDD \mathbf{B} are logically equivalent if for every assignment $A \in \mathcal{A}$, $A \models \varphi$ if and only if there is a path $\alpha \in \text{Path}^f(\mathbf{B})$ such that $A \in \mathcal{A}(\alpha)$.

For any CNF φ and OBDD \mathbf{B} , we use $\varphi \leq \mathbf{B}$ to denote that φ and \mathbf{B} are logically equivalent and for each path $\alpha \in \text{Path}^f(\mathbf{B})$ there is a clause $C \in \text{Cls}(\varphi)$ such that $\alpha \not\models C$.

2.3 OBDD Construction

The straightforward way to construct an OBDD is to start with a binary decision tree and then incrementally eliminate redundancies and identify identical subtrees. The other more efficient way follows the structure of the propositional formula. Such algorithms start with building OBDDs for variables or literals, and then construct more complex OBDDs by using OBDD operations for logical connectives.

Algorithm 1 presented below (from [16]) takes as an input two OBDDs \mathbf{B}_1 and \mathbf{B}_2 and returns their conjunction denoted by $\mathbf{B}_1 \wedge \mathbf{B}_2$. It proceeds from the root downward creating vertices in the resulting graph as follows:

1. the function `Dec` decomposes a non-terminal OBDD node into its constituent components, that is its variable and cofactors;
2. the function `Node` constructs a new OBDD node if it is not already present, and otherwise returns the already existent node.

Lemma 1 is a technical lemma which will be used to prove Corollary 1. It follows relatively straightforwardly from the definition of Algorithm 1.

Lemma 1. *Assume OBDDs \mathbf{B}_1 and \mathbf{B}_2 such that $\text{var}(\mathbf{B}_1) \neq \emptyset$ and $\text{var}(\mathbf{B}_2) \neq \emptyset$. Then Algorithm 1 returns the OBDD $\mathbf{B}_1 \wedge \mathbf{B}_2$ such that*

- for each $\alpha \in \text{Path}^f(\mathbf{B}_1 \wedge \mathbf{B}_2)$ there is $\beta \in \text{Path}^f(\mathbf{B}_1) \cup \text{Path}^f(\mathbf{B}_2)$ such that $\beta \subseteq \alpha$;
- for each $\beta \in \text{Path}^f(\mathbf{B}_1) \cup \text{Path}^f(\mathbf{B}_2)$ there is $\alpha \in \text{Path}^f(\mathbf{B}_1 \wedge \mathbf{B}_2)$ such that $\beta \subseteq \alpha$.

Proof. We give a proof by induction on $k = |\text{var}(\mathbf{B}_1) \cup \text{var}(\mathbf{B}_2)|$. For the basis step we choose $k = 0$ and the lemma trivially holds.

We assume that the lemma holds for any \mathbf{B}'_1 and \mathbf{B}'_2 such that $|\text{var}(\mathbf{B}'_1) \cup \text{var}(\mathbf{B}'_2)| \leq k - 1$. Let $|\text{var}(\mathbf{B}_1) \cup \text{var}(\mathbf{B}_2)| = k$ and $x \in \text{var}(\mathbf{B}_1) \cup \text{var}(\mathbf{B}_2)$ be the smallest variable. Then by the definition of an OBDD $|\text{var}(\text{high}(\mathbf{B}_1 \wedge \mathbf{B}_2))| \leq k - 1$ and $|\text{var}(\text{low}(\mathbf{B}_1 \wedge \mathbf{B}_2))| \leq k - 1$.

If $x \in \text{var}(\mathbf{B}_1)$ and $x \in \text{var}(\mathbf{B}_2)$ then by the definition of Algorithm 1 it returns the OBDD $\mathbf{B}_1 \wedge \mathbf{B}_2$ such that

- $\text{var}(\text{root}) = x$;
- $\text{high}(\mathbf{B}_1 \wedge \mathbf{B}_2) = \text{high}(\mathbf{B}_1) \wedge \text{high}(\mathbf{B}_2)$ and $\text{low}(\mathbf{B}_1 \wedge \mathbf{B}_2) = \text{low}(\mathbf{B}_1) \wedge \text{low}(\mathbf{B}_2)$.

```

Data:  $B_1, B_2$ 
Result:  $B_1 \wedge B_2$ 
if  $B_1 = \text{false}$  or  $B_2 = \text{false}$  then
  | return false;
end
if  $B_1 = \text{true}$  then
  | return  $B_2$ ;
end
if  $B_2 = \text{true}$  then
  | return  $B_1$ ;
end
 $(x, \text{high}(B_1), \text{low}(B_1)) = \text{Dec}(B_1);$ 
 $(y, \text{high}(B_2), \text{low}(B_2)) = \text{Dec}(B_2);$ 
if  $x = y$  then
  | return  $\text{Node}(x, \text{high}(B_1) \wedge \text{high}(B_2), \text{low}(B_1) \wedge \text{low}(B_2));$ 
end
if  $x \prec y$  then
  | return  $\text{Node}(x, \text{high}(B_1) \wedge B_2, \text{low}(B_1) \wedge B_2);$ 
end
if  $y \prec x$  then
  | return  $\text{Node}(y, B_1 \wedge \text{high}(B_2), B_1 \wedge \text{low}(B_2));$ 
end

```

Algorithm 1: The algorithm for constructing $B_1 \wedge B_2$

If $x \in \text{var}(B_1)$ and $x \notin \text{var}(B_2)$ then by the definition of Algorithm 1 it returns the OBDD $B_1 \wedge B_2$ such that

- $\text{var}(\text{root}) = x;$
- $\text{high}(B_1 \wedge B_2) = \text{high}(B_1) \wedge B_2$ and $\text{low}(B_1 \wedge B_2) = \text{low}(B_1) \wedge B_2.$

We use the induction hypothesis and conclude that the lemma holds.

The following corollary is a direct consequence of Lemma 1 and it will be used later to prove the main result.

Corollary 1. *Assume CNFs φ_1 and φ_2 and OBDDs B_1 and B_2 such that $\varphi_1 \preceq B_1$ and $\varphi_2 \preceq B_2$. Then Algorithm 1 returns the OBDD $B_1 \wedge B_2$ such that $\varphi_1 \wedge \varphi_2 \preceq B_1 \wedge B_2$.*

2.4 Reduction Rules

There are two reduction rules not affecting the semantics of OBDDs that can be used to reduce the size of the OBDDs constructed by Algorithm 1 defined as follows.

- **Merging:** If $\text{low}(p) \equiv \text{low}(q)$ and $\text{high}(p) \equiv \text{high}(q)$ for $p, q \in \text{Node}(B)$ then the node p can be removed. Any link to the node p is replaced by a link to the node q .
- **Elimination:** If $\text{low}(p) \equiv \text{high}(p)$ for $p \in \text{Node}(B)$ then the node p can be removed. Any link to the node p is replaced by a link to the root of $\text{high}(p)$. We write $B \rightarrow_p B'$ if B' is obtained from B by eliminating the node p .

We use B^\downarrow to denote the reduced OBDD obtained from B , that is no reduction rule can be applied to B^\downarrow any more.

Lemma 2 (Bryant [5]). *If B_1 and B_2 are logically equivalent OBDDs then $B_1^\downarrow \equiv B_2^\downarrow$.*

The total time complexity of the algorithm is $O(|B_1| \times |B_2|)$. In the worst case, the upper bound is achieved and $B_1 \wedge B_2$ can contain $O(|B_1| \times |B_2|)$ nodes.

Theorem 2 (Bryant [5]). *Let B_1 and B_2 be two reduced OBDDs, that is $B_1 \equiv B_1^\downarrow$ and $B_2 \equiv B_2^\downarrow$. Then the size of $B_1 \wedge B_2$ is $O(|B_1| \times |B_2|)$, and the number of merging and elimination steps to compute the reduced OBDD corresponding to $B_1 \wedge B_2$ is at most $|B_1 \wedge B_2|$.*

Proof. 1. By definition, any subOBDD of $B_1 \wedge B_2$ is of the form $B'_1 \wedge B'_2$, where B'_1 and B'_2 are subOBDDs of B_1 and B_2 respectively. Hence, the size of $B_1 \wedge B_2$ is bounded by $O(|B_1| \times |B_2|)$.

2. This is immediate since the elimination and merging rules strictly decrease the number of nodes.

2.5 Notations and Technical Background

Now we define notations that will be used in the rest of the paper and introduce some simple technical background we need to prove the main result. Thus, some additional properties related to the construction of an OBDD by Algorithm 1 are introduced in Lemma 3 and straightforward combinatorial results are defined in Lemma 4.

Let φ be a CNF and B be an OBDD such that $\varphi \sqsubseteq B$. We tacitly assume a function

$$F : \text{Path}^f(B) \rightarrow \text{Cls}(\varphi)$$

such that $\alpha \not\models F(\alpha)$ for an $\alpha \in \text{Path}^f(B)$. For each node $p \in \text{Node}(B)$, we define the set

$$\text{Cls}(p, \varphi) = \{C \in \text{Cls}(\varphi) \mid \exists \alpha \in \text{Path}^f(p) : C = F(\alpha)\}$$

For each $C \in \text{Cls}(\varphi)$ and $p \in \text{Node}(B)$, we define

$$\gamma(p, C) = |\{\alpha \in \text{Path}^f(p) \mid F(\alpha) = C\}| - 1$$

and

$$\tau(p, \varphi) = \sum_{C \in \text{Cls}(p, \varphi)} \gamma(p, C)$$

Suppose $\text{high}(p) \equiv \text{low}(p)$ and $B \rightarrow_p B'$ for a node $p \in \text{Node}(B)$ and an OBDD B' . The set $\text{Res}(p, \varphi)$ is defined as follows:

$$\text{Res}(p, \varphi) = \{\psi \in \text{Cnf} \mid \varphi \vdash_{\text{res}} \psi \text{ and } \varphi \wedge \psi \sqsubseteq B'\}$$

Let $x = \text{var}(p)$ and $q^h \in \text{Node}(\text{high}(p))$ and $q^l \in \text{Node}(\text{low}(p))$. We write

$$q^h \sim_p q^l$$

to denote that $\alpha.x.\beta \in \text{Path}^f(q^h)$ if and only if $\alpha.\neg x.\beta \in \text{Path}^f(q^l)$ for some α and β .

Let $B \rightarrow_p B'$. For simplicity, we define informally what we mean by $q \rightarrow_p q'$. The OBDD B' contains in fact the same nodes as B except the node p . We write

$$q \rightarrow_p q'$$

to denote that the node $q' \in \text{Node}(B')$ is in fact the node $q \in \text{Node}(B)$ after renaming as q' .

In the following we assume that clauses of φ are not subsumed while constructing the OBDD encoding it using Algorithm 1. This is formalised with Lemma 3 below.

Lemma 3. Assume a CNF φ and an OBDD B such that $\varphi \leq B$. Let $\text{high}(p) \equiv \text{low}(p)$ for some $p \in \text{Node}(B)$ with $x = \text{var}(p)$. Suppose the following holds:

1. $C \in \text{Cls}(\text{high}(p), \varphi)$ and $D \in \text{Cls}(\text{low}(p), \varphi)$;
2. Let $C = \neg x \vee C'$ and $D = x \vee D'$. Then for any literal l , if $l \in \text{Lit}(C')$ then $\neg l \notin \text{Lit}(D')$;
3. There is $x' \in \text{var}(C) \cap \text{var}(D)$ such that for any $x'' \in \text{var}(C) \cup \text{var}(D) \setminus \{x'\}$, $x'' \prec x'$.

Then for some α_1, α_2 there are $\alpha_1.x.\alpha_2 \in \text{Path}^f(\text{high}(p))$ and $\alpha_1.\neg x.\alpha_2 \in \text{Path}^f(\text{low}(p))$ such that

$$\alpha_1.x.\alpha_2 \not\models C$$

$$\alpha_1.\neg x.\alpha_2 \not\models D$$

Proof. In fact, the lemma statement is implied directly by the assumption $\text{high}(p) \equiv \text{low}(p)$ and the definition of Algorithm 1. But we will provide a somewhat formal proof by induction on $k = |\text{var}(\text{Path}^f(p))|$.

Let $k = 1$. Now we obtain that $\alpha_1 = \alpha_2 = \varepsilon$ where ε is the empty string and the lemma holds. We assume that the lemma holds for k with $k \geq 1$ and show that it hold for $k + 1$. We consider the following cases:

- The node p is not the root of B . Then the lemma holds by the induction hypothesis and the definition of Algorithm 1 for $\text{high}(\text{root}(B))$ and $\varphi|_{\neg \text{var}(\text{root}(B))}$ or for $\text{low}(\text{root}(B))$ and $\varphi|_{\text{var}(\text{root}(B))}$. Now the lemma holds straightforwardly for B and φ .
- The node p is the root of B . Let $q^h = \text{root}(\text{high}(p))$ and $q^l = \text{root}(\text{low}(p))$. It follows from $\text{high}(p) \equiv \text{low}(p)$ that $\text{high}(q^h) \equiv \text{high}(q^l)$ and $\text{low}(q^h) \equiv \text{low}(q^l)$.

We construct the OBDD B^h by redirecting there true and false branches of p to the $\text{root}(\text{high}(q^h))$ and $\text{root}(\text{high}(q^l))$ correspondingly; and B^l by redirecting there true and false branches of p to the $\text{root}(\text{low}(q^h))$ and $\text{root}(\text{low}(q^l))$ correspondingly. Let $y = \text{root}(q^h) = \text{root}(q^l)$. By the induction hypothesis the lemma holds for B^h and $\varphi|_{\neg y}$ or for B^l and $\varphi|_y$. Hence, the lemma holds for arbitrary B and φ .

Lemma below presents some technical results which will be used to prove Theorem 6.

Lemma 4. Let S be a finite set such that $|S| > 0$, and $B_1, \dots, B_l \subseteq S$ be a sequence with:

- $B_l = S$
- For each B_i , $1 \leq i < l$, one of the following holds:
 - $B_i = \{s\}$ for $s \in S$
 - $B_i = B_j \cup B_k$ for some j, k with $j < k < i$

Then $l = 2|S| - 1$.

Proof. We give a proof by induction on $|S|$. As the basis step we choose $|S| = 1$. Then the lemma hold as trivially $l = 1$. Let the lemma hold for any $|S|$. Assume a set S' such that $|S'| = |S| + 1$. Then $l' = l + 2 = (2|S| - 1) + 2 = 2|S'| - 1$.

3 OBDDs and Resolution as Proof Systems

Proof systems based on resolution and OBDDs are so-called refutational proof systems. A refutation of an unsatisfiable CNF φ starts with the clauses of φ and derives a contradiction represented by the empty clause \perp for resolution and by the false node for OBDDs.

Any proof system operating with OBDDs can be seen as an instance of the constraints based proof system. In the following we consider the OBDD proof system which uses two rules, Axiom and Join.

Definition 3 (OBDD refutation). An OBDD refutation of a CNF φ is a sequence of OBDDs B_1, \dots, B_k such that the following holds:

- Axiom : $B_i \equiv C_i$ with $1 \leq i \leq |\text{Cls}(\varphi)|$;
- Join : $B_i \equiv B_{j'}^\downarrow \wedge B_{j''}^\downarrow$ with $1 \leq j' < j'' < i$ and $|\text{Cls}(\varphi)| < i \leq k$;
- $B_k^\downarrow \equiv \text{false}$.

The size of the OBDD refutation is defined as $\sum_{i=1}^k |B_i|$.

Without loss of generality we can assume that each OBDD is used exactly once, that is if a CNF φ consists of m clauses then the number of OBDDs in the OBDD refutation of φ is exactly $2m - 1$.

The resolution proof system goes back to Robinson [15] and consists of a single rule. It derives from two clauses $l \vee C$ and $\neg l \vee D$, such that C and D do not contain a complementary literal, the new clause $C \vee D$ called the resolvent of $l \vee C$ and $\neg l \vee D$, and denoted in the following by $\text{res}(l \vee C, \neg l \vee D)$.

When we write $\text{res}(C, D)$, we assume that there is a literal l such that $l \in \text{Lit}(C)$ and $\neg l \in \text{Lit}(D)$; moreover, the clauses C and D contain no other complementary literals.

Definition 4 (Resolution refutation). A resolution refutation of a CNF φ of size k is a sequence of clauses C_1, \dots, C_k such that

1. Axiom : $C_i \in \text{Cls}(\varphi)$ with $1 \leq i \leq |\text{Cls}(\varphi)|$;
2. Join : $C_i = \text{res}(C_{j'}, C_{j''})$ with $1 \leq j' < j'' < i$ and $|\text{Cls}(\varphi)| < i \leq k$;
3. $C_k = \perp$.

We say that k is the size of the resolution refutation. We write $\varphi \vdash_{\text{res}} \psi$ for any CNF ψ such that $\psi = \bigwedge_{i=|\text{Cls}(\varphi)|+1}^{k'} C_i$ with $k' \leq k$.

4 Simulating OBDDs by Resolution

In the rest of the paper we show formally that if there is an OBDD refutation of a CNF φ of size n then there is a resolution refutation of φ of size at most n^2 . The existence of such resolution refutation is based on the following observations:

1. elimination of a node can be simulated by at most $|\text{Cls}(\varphi)|$ resolution steps;
2. $|\text{Cls}(\varphi)| \leq n$;
3. the number of nodes which can be removed by the elimination rule is at most n .

Our main argument is based on the idea that the elimination rule can be simulated by applying the resolution rule on the variable corresponding to the eliminated node [11].

We strengthen this idea and prove that the number of resolution steps corresponding to the elimination of a node is bounded by the number of clauses in the input CNF φ .

Moreover, we show that it is an invariant property: although resolution steps generate new clauses, the number of resolution steps needed to simulate elimination of a node in an intermediate OBDD remains bounded by the number of clauses of φ encoded by this OBDD. We also show that it is sufficient to simulate only the elimination rule. As the merging rule plays no role in the context, we assume for simplicity that all intermediate OBDDs are merged.

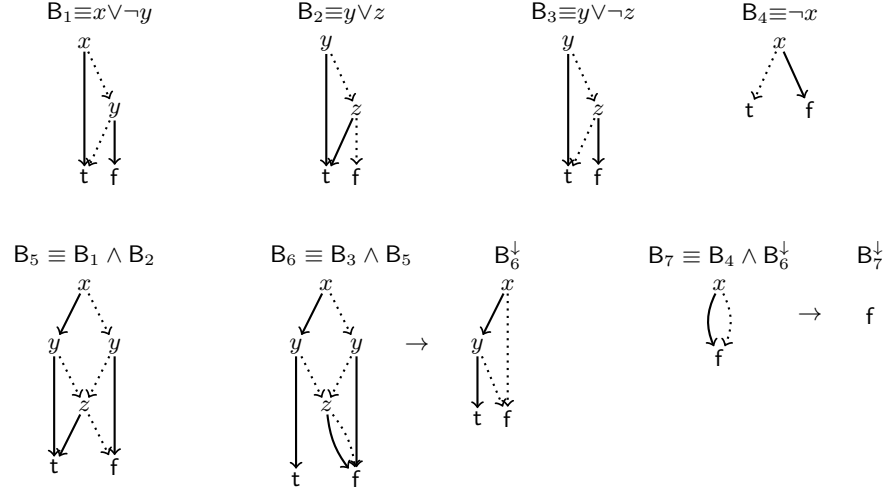


Fig. 1. The OBDD refutation of $\varphi = (x \vee \neg y) \wedge (y \vee z) \wedge (y \vee \neg z) \wedge \neg x$ where the solid lines represent the true-branches and the dotted lines represent the false-branches

Example 1. Before giving technical details, we provide a simple illustrating example. We consider the CNF

$$\varphi = (x \vee \neg y) \wedge (y \vee z) \wedge (y \vee \neg z) \wedge \neg x$$

An OBDD refutation of φ is depicted in Figure 1. The OBDDs B_6^\downarrow and B_7^\downarrow are obtained from B_6 and B_7 correspondingly by applying the elimination rule. Consider the following resolution refutation of φ :

$$C_1 = x \vee \neg y, C_2 = y \vee z, C_3 = y \vee \neg z, C_4 = \neg x,$$

$$C_5 = \text{res}(C_2, C_3) = y, C_6 = \text{res}(C_1, C_5) = x, C_7 = \text{res}(C_4, C_6) = \perp$$

It simulates the OBDD refutation as follows:

- By resolving the clauses C_2 and C_3 we eliminate the occurrences of z . By resolving the clauses C_1 and C_5 we eliminate the occurrences of y . The new set of clauses decodes the OBDD B_6^\downarrow .
- By resolving the clauses C_4 and C_6 we eliminate the occurrences of x in B_7 .

4.1 Simulation of the Elimination Rule

In this section we show that elimination of a node can be simulated by at most $|\text{Cls}(\varphi)|$ resolution steps, where φ is the input unsatisfiable CNF.

Lemma 5 demonstrates that elimination of a node p can be simulated by at most $k/2$ resolution steps, where k is the number of paths going through p to the false-node. This is a variant of Lemma 2 from [11] which serves our needs better.

Lemma 5. *Assume a CNF φ and an OBDD B such that $\varphi \trianglelefteq B$. Let $B \rightarrow_p B'$ for a node $p \in \text{Node}(B)$ and an OBDD B' . Then there is a CNF $\psi \in \text{Res}(p, \varphi)$ such that*

$$|\text{Cls}(\psi)| \leq |\text{Path}^f(p)|/2$$

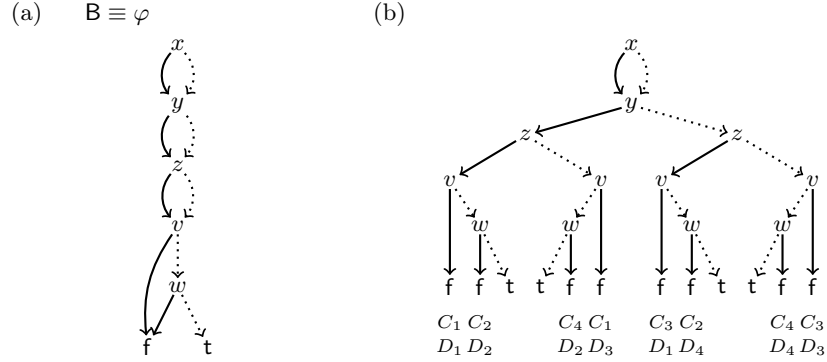


Fig. 2. An example illustrating Theorem 5: (a) the OBDD encoding of φ ; and (b) the mapping of the false-paths of the OBDD onto the set of clauses of φ

Proof. Assume that $x = \text{var}(p)$. Let ψ be the smallest CNF satisfying the following: for all α and β such that $\alpha.x.\beta, \alpha.\neg x.\beta \in \text{Path}^f(p)$,

$$\neg(\exists C \in \text{Cls}(\varphi) : \alpha.\beta \not\models C) \implies \text{res}(F(\alpha.x.\beta), F(\alpha.\neg x.\beta)) \in \text{Cls}(\psi)$$

By construction, $\psi \in \text{Res}(p, \varphi)$ and $|\text{Cls}(\psi)| \leq |\text{Path}^f(p)|/2$.

Example 2. We provide another illustrating example. Consider the CNF φ consisting of the following eight clauses:

$$\begin{aligned} C_1 &= \neg x \vee \neg y \vee \neg v & C_2 &= \neg x \vee \neg z \vee \neg w & C_3 &= \neg x \vee y \vee \neg v & C_4 &= \neg x \vee z \vee \neg w \\ D_1 &= x \vee \neg z \vee \neg v & D_2 &= x \vee \neg y \vee \neg w & D_3 &= x \vee z \vee \neg v & D_4 &= x \vee y \vee \neg w \end{aligned}$$

Figure 2 represents the OBDD encoding of φ , and the mapping of the false-paths onto the clauses of φ . Elimination of the node labelled with x can be simulated by the following resolution steps:

$$\begin{aligned} \text{res}(C_1, D_1) &= \neg y \vee \neg z \vee \neg v & \text{res}(C_2, D_2) &= \neg y \vee z \vee \neg v & \text{res}(C_1, D_3) &= \neg y \vee \neg z \vee \neg w \\ \text{res}(C_4, D_2) &= \neg y \vee z \vee \neg w & \text{res}(C_3, D_1) &= y \vee \neg z \vee \neg v & \text{res}(C_2, D_4) &= \vee z \vee \neg v \\ \text{res}(C_3, D_3) &= y \vee \neg z \vee \neg w & \text{res}(C_4, D_4) &= y \vee z \vee \neg w \end{aligned}$$

Corollary 2 below follows directly from Theorem 5.

Corollary 2. Assume an OBDD B and a CNF φ such that $\varphi \sqsubseteq B$. Suppose $\text{high}(p) \equiv \text{low}(p)$ for some $p \in \text{Node}(B)$. Let $P^h \subseteq \text{Path}^f(\text{high}(p))$ and $P^l \subseteq \text{Path}^f(\text{low}(p))$ be sets such that $\alpha.x.\beta \in P^h$ if and only if $\alpha.\neg x.\beta \in P^l$ for some α and β . Suppose $\bar{P}^h = \text{Path}^f(\text{high}(p)) \setminus P^h$ and $\bar{P}^l = \text{Path}^f(\text{low}(p)) \setminus P^l$. Suppose for each pair of paths $(\alpha.x.\beta, \alpha.\neg x.\beta) \in \bar{P}^h \times \bar{P}^l$ there is a pair of paths $(\alpha'.x.\beta', \alpha'.\neg x.\beta') \in P^h \times P^l$ and clauses $C, D \in \text{Cls}(\varphi)$ such that

$$\alpha.x.\beta, \alpha'.x.\beta' \not\models C$$

$$\alpha.\neg x.\beta, \alpha'.\neg x.\beta' \not\models D$$

Let $k = |\bar{P}^h|$. Then there is $\psi \in \text{Res}(p, \varphi)$ such that

$$|\text{Cls}(\psi)| \leq |\text{Path}^f(p)|/2 - k$$

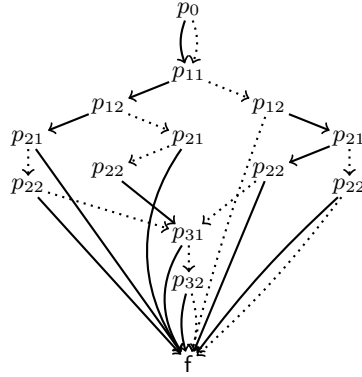


Fig. 3. The OBDD encoding of $\overline{\text{PHP}}_2$ with $\text{high}(p_0) \equiv \text{low}(p_0)$

Example 3. The formulas PHP_n , $n \geq 1$, encoding the pigeonhole principle were studied intensively in relation to complexity of different propositional proof systems and they are defined as follows.

$$\text{PHP}_n = \bigwedge_{i=1}^{n+1} \bigvee_{j=1}^n p_{ij} \wedge \bigwedge_{\substack{1 \leq i < j \leq n+1 \\ 1 \leq k \leq n}} \neg p_{ik} \vee \neg p_{jk}$$

We build the formulas $\overline{\text{PHP}}_n$ by doubling the number of clauses of PHP_n : for some new variable p_0

$$p_0 \vee C, \neg p_0 \vee C \in \text{Cls}(\overline{\text{PHP}}_n)$$

if and only if $C \in \text{Cls}(\text{PHP}_n)$.

Let B be the OBDD encoding $\overline{\text{PHP}}_2$ as it is depicted in Figure 3. Elimination of the node p_0 can be trivially simulated by $|\text{Cls}(\text{PHP}_2)|$ resolution steps. It is sufficient to add the resolvent $\text{res}(p_0 \vee C, \neg p_0 \vee C) = C$.

While the size of the OBDDs encoding $\overline{\text{PHP}}_n$ will grow exponentially in n , elimination of the node labelled with p_0 can be simulated in the same manner by the number of resolution steps bounded by $|\text{Cls}(\overline{\text{PHP}}_n)|$ which grows polynomially in n .

The subsequent statements improve the upper bound on the number of resolution steps needed to simulate elimination of an arbitrary node. Namely, we show that the number of resolution steps sufficient to simulate elimination of a node p is bounded by $|\text{Cls}(\varphi)|$.

Lemma 6. *Let φ be a CNF and B be an OBDD such that $\varphi \sqsubseteq B$. Suppose $B \rightarrow_p B'$ for some $p \in \text{Node}(B)$ and B' . Then there is a CNF $\psi \in \text{Res}(p, \varphi)$ such that*

$$|\text{Cls}(\psi)| \leq |\text{Cls}(\varphi)|$$

Proof. If $|\text{Cls}(\varphi)| \geq |\text{Path}^f(p)|/2$ then the theorem holds by Theorem 5. We assume that

$$|\text{Cls}(\varphi)| < |\text{Path}^f(p)|/2$$

Let $m = |\text{Path}^f(p)|/2$ and $m' = |\text{Cls}(\varphi)|$ with $m' - m > 0$; and $p^h = \text{root}(\text{high}(p))$, $p^l = \text{root}(\text{low}(p))$ and $x = \text{var}(p)$. Let P be the following set

$$P = \{(\alpha.x.\beta, \alpha.\neg x.\beta) \mid \exists \alpha, \beta : \alpha.x.\beta \in \text{Path}^f(\text{high}(p)), \alpha.\neg x.\beta \in \text{Path}^f(\text{low}(p))\}$$

Assume a function $F' : P \rightarrow \text{Cls}(\varphi)$ such that $F'((\alpha.x.\beta, \alpha.\neg x.\beta)) = C$ if $\alpha.x.\beta \not\models C$ or $\alpha.\neg x.\beta \not\models C$ and $\bigcup_{(\alpha.x.\beta, \alpha.\neg x.\beta) \in P} F'((\alpha.x.\beta, \alpha.\neg x.\beta)) = \text{Cls}(p, \varphi)$.

1. We define the sets S, S_1, \dots, S_l with $l = 2m - 1$ as follows:

- $S = P$;
- $S_i = \{s_i\}$ for $s_i \in S'$ with $1 \leq i \leq m$;
- $S_i = S'_j \cup S'_k$ with $i > j > k$ and $m + 1 \leq i \leq 2m - 1$.

2. We define the sets S', S'_1, \dots, S'_l with $l = 2m - 1$ as follows:

- $S' = \text{Cls}(p^h, \varphi)$;
- $S'_i = \{s'_i\}$ for $s'_i \in S'$ with $1 \leq i \leq m$;
- $S'_i = S'_j \cup S'_k$ with $i > j > k$ and $m + 1 \leq i \leq 2m - 1$.

Moreover, we assume that $S'_i = \bigcup_{(\alpha.x.\beta, \alpha.\neg x.\beta) \in S_i} F'((\alpha.x.\beta, \alpha.\neg x.\beta)) \cap \text{Cls}(p^h, \text{Cls}\varphi)$.

3. We define the sets S'', S''_1, \dots, S''_l with $l = 2m - 1$ as follows:

- $S'' = \text{Cls}(p^l, \varphi)$;
- $S''_i = \{s''_i\}$ for $s''_i \in S''$ with $1 \leq i \leq m$;
- $S''_i = S''_j \cup S''_k$ with $i > j > k$ and $m + 1 \leq i \leq 2m - 1$.

Moreover, we assume that $S''_i = \bigcup_{(\alpha.x.\beta, \alpha.\neg x.\beta) \in S_i} F'((\alpha.x.\beta, \alpha.\neg x.\beta)) \cap \text{Cls}(p^l, \text{Cls}\varphi)$.

Let $A_i = S'_i \cup S''_i$ with $1 \leq i \leq 2m - 1$ and $\mathcal{A} = \{A_1, \dots, A_{2m-1}\}$. Now it follows from Lemma 4 and the definition of an OBDD that there is a set $\overline{\mathcal{A}} = \{A_{i_1}, \dots, A_{i_{m-m'}}\}$ such that for each $\overline{A} \in \overline{\mathcal{A}}$ there is $A \in \mathcal{A}$ such that $\overline{A} \subseteq A$. Hence, by Lemma 3 there are $m - m'$ pairs

$$(\alpha_1.x.\beta_1, \alpha_1.\neg x.\beta_1), \dots, (\alpha_m.x.\beta_m, \alpha_m.\neg x.\beta_m)$$

in the set P , let us call this set P' , such that for each $(\alpha'.x.\beta', \alpha'.\neg x.\beta') \in P'$ there is $(\alpha.x.\beta, \alpha.\neg x.\beta) \in P \setminus P'$ such that $\alpha.x.\beta, \alpha'.x.\beta' \not\models C$ and $\alpha.\neg x.\beta, \alpha'.\neg x.\beta' \not\models D$ for some $C, D \in \text{Cls}(\varphi)$. By Corollary 2 we obtain that there is a CNF $\psi \in \text{Res}(p, \varphi)$ such that $|\text{Cls}(\psi)| \leq |\text{Cls}(\varphi)|$.

It follows from Lemmas 5 and 6 that the number of resolution steps needed to simulate elimination of node p is bounded by the minimum of $|\text{Path}^f(p)|/2$ and $|\varphi|$.

Lemma 7 below is a straightforward consequence of Lemma 6, and it somewhat relates the upper bound on the number of resolution steps simulating elimination of p and the number of the false-paths that go through the node p in combination with the number of the clauses falsifying these paths (expressed by $\tau(., .)$).

Lemma 7. *Let φ be a CNF and B be an OBDD such that $\varphi \leq B$. Suppose $B \rightarrow_p B'$ for some $p \in \text{Node}(B)$ and B' . Then there is a $\psi \in \text{Res}(p, \varphi)$ such that*

$$|\text{Cls}(\psi)| \leq |\text{Path}^f(p)| - \tau(p, \varphi)$$

Proof. We take into account that by definition of $\tau(., .)$, $|\text{Path}^f(p)| - \tau(p, \varphi) = |\text{Cls}(p, \varphi)|$, and the lemma trivially holds.

4.2 Invariant

Now we will prove that although resolution steps generate new clauses, the number of resolution steps needed to simulate elimination of a node remains bounded by the number of clauses encoded by this OBDD. In fact, we demonstrate a kind of monotonicity expressed by Lemma 8.

Lemma 8. *Assume a CNF φ and an OBDD B with $\varphi \leq B$. Suppose $B \rightarrow_p B'$ and $q \rightarrow_p q'$ for $p, q \in \text{Node}(B)$ and $q' \in \text{Node}(B')$. Let $\text{high}(q') \equiv \text{low}(q')$. Then there is a $\varphi' \in \text{Res}(p, \varphi)$ such that*

$$|\text{Path}^f(q')| - \tau(q', \varphi \wedge \varphi') \leq |\text{Path}^f(q)| - \tau(q, \varphi)$$

Proof. 1. Suppose the nodes p and q are not connected by a path. Then removing the node p does not affect the false-paths that go through the node q . That is, $\text{Path}^f(q') = \text{Path}^f(q)$ and $\tau(q', \varphi \wedge \varphi') = \tau(q, \varphi)$. Hence,

$$|\text{Path}^f(q')| - \tau(q', \varphi \wedge \varphi') = |\text{Path}^f(q')| - \tau(q', \varphi) = |\text{Path}^f(q)| - \tau(q, \varphi)$$

Now we assume that p and q are connected by a path. We construct φ' as it is defined in the proof of Lemma 5 and we consider following cases.

2. Let $q \in \text{Node}(B^p)$ where B^p is the subOBDD of B rooted at the node p .

We observe that

$$\alpha.\beta \not\models \text{res}(C, D)$$

if and only if $\alpha.x.\beta \not\models C$ and $\alpha.\neg x.\beta \not\models D$ for some $\alpha.\beta \in \text{Path}^f(\text{high}(p))$ (alternatively, we could use $\text{Path}^f(\text{low}(p))$ as $\text{high}(p) \equiv \text{low}(p)$) and $C \in \text{Cls}(\text{root}(\text{high}(p)), \varphi)$, $D \in \text{Cls}(\text{root}(\text{low}(p)), \varphi)$. Hence,

$$|\text{Path}^f(q')| - \tau(q', \varphi \wedge \varphi') \leq |\text{Path}^f(q)| - \tau(q, \varphi)$$

3. Let $p \in \text{Node}(B^q)$ where B^q is the subOBDD of B rooted at the node q .

Let $\text{Path}^f(p) = \text{Path}^f(q) \setminus \text{Path}^f(p)$. Let

$$\mathcal{C}^p = \{C \in \text{Cls}(p, \varphi) \mid \neg \exists \alpha \in \text{Path}^f(q) \setminus \text{Path}^f(p) : \alpha \not\models C\}$$

$$\mathcal{C}^q = \{C \in \text{Cls}(q, \varphi) \mid \neg \exists \alpha \in \text{Path}^f(p) : \alpha \not\models C\}$$

$$\mathcal{C}^{q.p} = \{C \in \text{Cls}(q, \varphi) \mid \exists \alpha \in \text{Path}^f(p), \beta \in \text{Path}^f(q) \setminus \text{Path}^f(p) : \alpha, \beta \not\models C\}$$

We use the same arguments as in case 1 for the clauses in \mathcal{C}^q , the same arguments as in case 2 for the clauses in \mathcal{C}^p and apply Lemma 3 for the clauses in $\mathcal{C}^{q.p}$.

Lemma 9. *Assume CNFs φ_1 and φ_2 , and OBDDs B_1 and B_2 with $\varphi_1 \leq B_1$ and $\varphi_2 \leq B_2$. Let for any $q_1 \in \text{Node}(B_1)$ and $q_2 \in \text{Node}(B_2)$, and some $k_1, k_2 \geq 0$*

- $|\text{Path}^f(q_1)| - \tau(q_1, \varphi_1) \leq k_1$
- $|\text{Path}^f(q_2)| - \tau(q_2, \varphi_2) \leq k_2$

Then Algorithm 1 returns the OBDD $B_1 \wedge B_2$ such that for any $q \in \text{Node}(B_1 \wedge B_2)$

$$|\text{Path}^f(q)| - \tau(q, \varphi_1 \wedge \varphi_2) \leq k_1 + k_2$$

Proof. Observe that for any CNF φ and an OBDD B such that $\varphi \leq B$ and $q \in \text{Node}(B)$,

$$|\text{Path}^f(q)| - \tau(q, \varphi) \leq |\text{Path}^f(B)| - \tau(\text{root}(B), \varphi)$$

We recall that by Lemma 1, $\varphi_1 \wedge \varphi_2 \leq B_1 \wedge B_2$ and define the sets S_1 and S_2 as follows:

- $S_1 = \{\alpha \in \text{Path}^f(B_1 \wedge B_2) \mid F(\alpha) \in \text{Cls}(\varphi_1)\};$
- $S_2 = \{\alpha \in \text{Path}^f(B_1 \wedge B_2) \mid F(\alpha) \in \text{Cls}(\varphi_2)\}.$

That is, the set S_1 contains the false-paths of $B_1 \wedge B_2$ falsified by the clauses of φ_1 and S_2 contains the false-paths of $B_1 \wedge B_2$ falsified by the clauses of φ_2 . Suppose

- $m_1 = |S_1| - |\text{Path}^f(\text{root}(B_1))|;$
- $m_2 = |S_2| - |\text{Path}^f(\text{root}(B_2))|.$

It follows from Lemma 1 and the definition of $\tau(.,.)$ that

$$\tau(\text{root}(B_1 \wedge B_2), \varphi_1 \wedge \varphi_2) = \tau(\text{root}(B_1), \varphi_1) + \tau(\text{root}(B_2), \varphi_2) + m_1 + m_2$$

and therefore for any $q \in \text{Node}(\varphi_1 \wedge \varphi_2)$

$$\begin{aligned} |\text{Path}^f(q)| - \tau(q, \varphi_1 \wedge \varphi_2) &\leq |\text{Path}^f(B_1 \wedge B_2)| - \tau(\text{root}(B_1 \wedge B_2), \varphi_1 \wedge \varphi_2) \\ &= (|\text{Path}^f(B_1)| + |\text{Path}^f(B_2)| + m_1 + m_2) - \\ &\quad (\tau(\text{root}(B_1), \varphi_1) + \tau(\text{root}(B_2), \varphi_2) + m_1 + m_2) \\ &\leq k_1 + k_2 \end{aligned}$$

Now we combine the results established by Lemmas 6-9 and obtain the following corollary.

Corollary 3. *Assume an unsatisfiable CNF φ . Let B_1, \dots, B_k be an OBDD refutation of φ . Then elimination of a node in any OBDD B_i , $1 \leq i \leq k$, can be simulated by at most $|\text{Cls}(\varphi)|$ resolution steps.*

Example 4. The CNFs PHP_n , $n \geq 1$, formalising the pigeonhole principle is presented in Example 3. We consider the OBDD refutation of PHP_2 depicted in Figure 4.

The following resolution refutation simulates removing the nodes of the OBDD B_{17} .

$$\begin{aligned} p_{32} : \quad & \neg p_{12} \vee p_{31} = \text{res}(\neg p_{12} \vee \neg p_{32}, p_{31} \vee p_{32}) \\ & \neg p_{22} \vee p_{31} = \text{res}(\neg p_{22} \vee \neg p_{32}, p_{31} \vee p_{32}) \end{aligned}$$

$$\begin{aligned} p_{31} : \quad & \neg p_{11} \vee \neg p_{12} = \text{res}(\neg p_{11} \vee \neg p_{31}, \neg p_{12} \vee p_{31}) \\ & \neg p_{12} \vee \neg p_{21} = \text{res}(\neg p_{21} \vee \neg p_{31}, \neg p_{12} \vee p_{31}) \\ & \neg p_{11} \vee \neg p_{22} = \text{res}(\neg p_{11} \vee \neg p_{31}, \neg p_{22} \vee p_{31}) \\ & \neg p_{21} \vee \neg p_{22} = \text{res}(\neg p_{21} \vee \neg p_{31}, \neg p_{22} \vee p_{31}) \end{aligned}$$

$$\begin{aligned} p_{22} : \quad & \neg p_{11} \vee p_{21} = \text{res}(p_{21} \vee p_{22}, \neg p_{11} \vee \neg p_{22}) \\ & \neg p_{12} \vee p_{21} = \text{res}(p_{21} \vee p_{22}, \neg p_{12} \vee \neg p_{22}) \end{aligned}$$

$$\begin{aligned} p_{21} : \quad & \neg p_{11} = \text{res}(\neg p_{11} \vee \neg p_{21}, \neg p_{11} \vee p_{21}) \\ & \neg p_{12} = \text{res}(\neg p_{12} \vee \neg p_{21}, \neg p_{12} \vee p_{21}) \end{aligned}$$

$$p_{12} : \quad p_{11} = \text{res}(p_{11} \vee p_{12}, \neg p_{12})$$

$$p_{11} : \quad \perp = \text{res}(p_{11}, \neg p_{11})$$

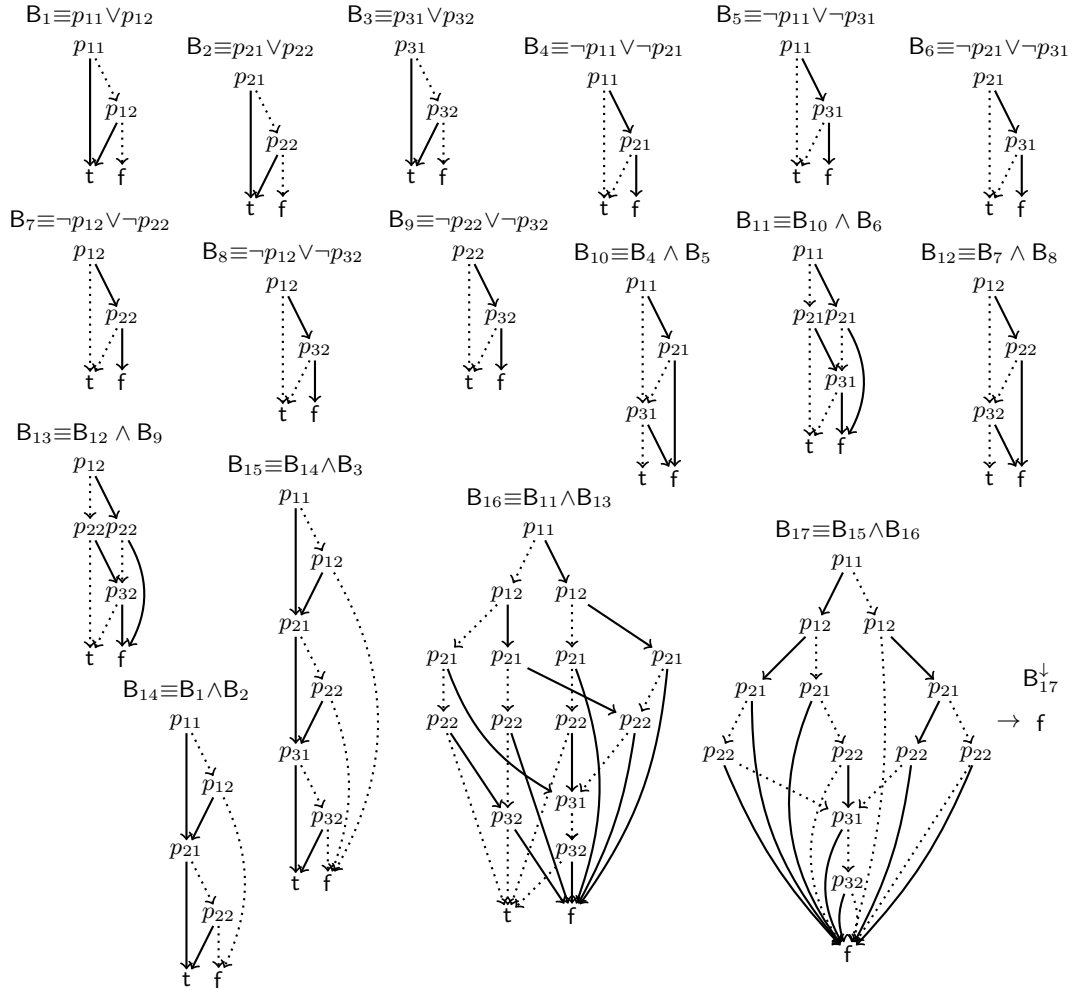


Fig. 4. The OBDD refutation of PHP_2

5 The Main Result

Now we establish the main result that any OBDD refutation of an unsatisfiable CNF φ of size n can be simulated by a resolution refutation of φ of size at most $O(n^2)$.

Theorem 3. *Assume an unsatisfiable CNF φ . If there is an OBDD refutation of φ of size n then there is a resolution refutation of it of size $O(|\text{Cls}(\varphi)| \cdot n)$.*

Proof. By Corollary 3, elimination of a node can be simulated by at most $|\text{Cls}(\varphi)|$ steps. Since the OBDD refutation has size n , we obtain that there is a resolution refutation of φ of size $O(|\text{Cls}(\varphi)| \cdot n)$.

Now, Theorem 1 stating that if there is an OBDD refutation of φ of size n then there is a resolution refutation of φ of size $O(n^2)$ follows straightforwardly from Theorem 3.

Proof. (Proof of Theorem 1) We can assume without loss of generality that φ is a minimally unsatisfiable CNF, that is removing any clause from φ will result in a satisfiable CNF.

Then the size of any OBDD refutation of φ is at least $|\text{Cls}(\varphi)|$, that is $|\text{Cls}(\varphi)| \leq n$. By Theorem 3, there is a resolution refutation of φ of size $O(k)$, where

$$k \leq |\text{Cls}(\varphi)| \cdot n \leq n^2$$

6 Conclusions

The main reason for this study comes from the interest in providing theoretical explanations of the relative efficiency of algorithms used in SAT solving.

In this paper we show that resolution simulates OBDDs polynomially if we limit both to CNFs and thus answered the open question of Groote and Zantema posed in [8] whether there exists unsatisfiable CNFs having polynomial OBDD refutations and exponentially long resolution refutations.

The goal of this study was to show the existence of such a polynomial simulation rather than provide the tightest upper bound. We envisage that an OBDD refutation can be simulated by resolution refutation with linear increase in size but it would require a more elaborated proof.

Acknowledgments

The author thanks Erika Ábrahám for helpful discussions at the early stage of this study.

References

1. S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, June 1978.
2. A. Atserias, P. Kolaitis, and M. Vardi. Constraint propagation as a proof system. In *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *LNCS*, pages 77–91, 2004.
3. M. Bonet, T. Pitassi, and R. Raz. On interpolation and automatization for Frege systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000.
4. R. T. Boute. The binary decision machine as a programmable controller. *EUROMICRO Newsletter*, 1(2):16–22, January 1976.
5. R. Bryant. Graph-based algorithms for boolean function manipulation. 8(C-35):677–691, 1986.
6. S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.
7. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
8. J. F. Groote and H. Zantema. Resolution and binary decision diagrams cannot simulate each other polynomially. *Discrete Applied Mathematics*, 130(2):157–171, 2003.
9. J. Krajíček. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *The Journal of Symbolic Logic*, 73(1):227–237, 2008.
10. C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, 38:985–999, 1959.
11. N. Peltier. Extended resolution simulates binary decision diagrams. *Discrete Applied Mathematics*, 156(6):825–837, 2008.
12. K. Pipatsrisawat and A. Darwiche. On the power of clause-learning SAT solvers with restarts. In *Principles and Practice of Constraint Programming*, 2009.

13. K. Pipatsrisawat and A. Darwiche. On modern clause-learning satisfiability solvers. *J. Autom. Reasoning*, 44(3):277–301, 2010.
14. A. Razborov. Propositional proof complexity. *J. ACM*, 50(1):80–82, 2003.
15. J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.
16. C. Sinz and A. Biere. Extended resolution proofs for conjoining BDDs. In *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia*, volume 3967 of *LNCS*. Springer, 2006.
17. O. Tveretina, C. Sinz, and H. Zantema. Ordered binary decision diagrams, pigeonhole formulas and beyond. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:35–58, 2010.
18. T. E. Uribe and M. E. Stickel. Ordered binary decision diagrams and the davis-putnam procedure. In *First International Conference on Constraints in Computational Logics*, volume 845, pages 34–49. Lecture Notes in Computer Science, 1994.
19. I. Wegener. *Branching programs and binary decision diagrams: theory and applications*. Society for Industrial and Applied Mathematics, 2000.